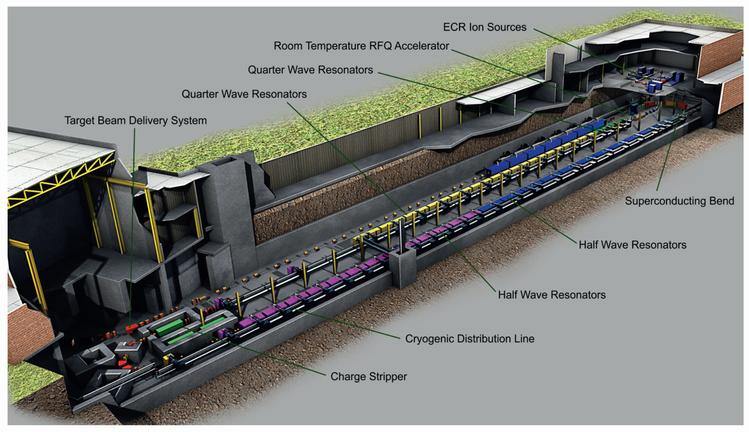
EPICS Support Module for Efficient UDP Communication With FPGAs

Martin Konrad, Enrique Bernal, Mark Davis
Facility for Rare Isotope Beams (FRIB), Michigan State University, East Lansing, MI 48824 USA

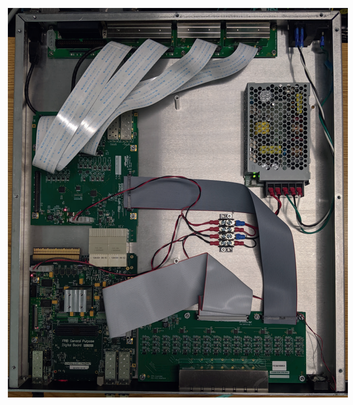
FRIB

- New heavy ion accelerator facility for nuclear physics
- Driver linac is designed to accelerate all stable ions to energies >200 MeV/u
- Beam power on target is up to 400 kW
- 350 cavities, 300 IOCs, thousands of devices



Hardware and Requirements

- Used with two in-house developed systems based on a low-cost pizza-box design with a Spartan 6 FPGA
  - ~350 low-level RF controllers
  - ~55 machine-protection nodes
  - → Driver needs to be very reliable
- Soft-core runs a C program which handles Ethernet communication
  - No operating system, no IP stack → need to use UDP
- LLRF firmware is improved continuously, registers are added/modified frequently
  → Driver needs to be flexible



Design Decisions

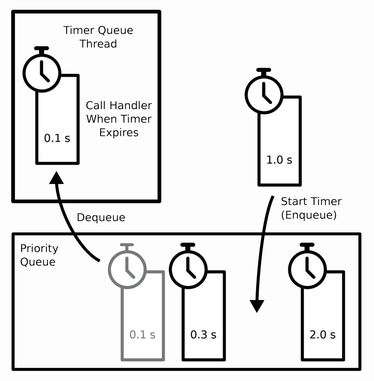
- AsynPortDriver is used to simplify implementation of asynchronous operations
- Asyn parameters are dynamically created while loading the IOC database
  - Register addresses and data types (signed int, unsigned int, fixed point 16.16 etc.) are read from INP/OUT fields
  - Strings are parsed using std::regex (C++11)
- Registers are read out periodically as a large array rather than transferring them one by one to improve efficiency
- Timers are leveraged for implementing operations not supported by Asyn

UDP Protocol

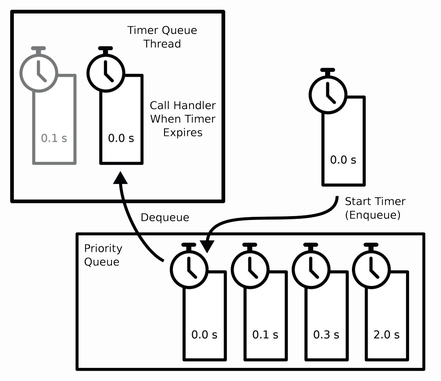
- IOC initiates communication by sending a UDP request package, the device generally responds with one or more UDP packages
- Specified commands:
  - Read registers
  - Write registers
  - Read waveform
  - Read a block from persistent memory
  - Erase a block of persistent memory
  - Write a block to persistent memory
  - Request write access
- Three different memory regions are defined for registers
  - Read-only memory (e. g. read out a sensor)
  - Read/write memory (e. g. read/write set-point)
  - “Write-once” memory (for commands triggering some action like “start ramp”)
- Multiple clients can read from the device at the same time but only one of them can have write access
  - Session ID in each package helps to distinguish clients
  - Write access expires when not used for a while

Support Module is Timer Driven

- Timers are used to perform the following operations
  - Periodic read-out of registers (10 Hz)
  - Incremental array read/write without blocking other communication
  - Detecting communication problems (timeouts)
  - Periodically sending a request to keep write access (not needed when writes are performed)
- An instance of epicsTimerQueue is managing the timers
  - Timers are stored in a priority queue (ordered by expiration time)
  - The timer-queue thread waits until the first timer in the queue expires before calling its handler
- Each device has its own timer queue resulting in one additional thread per device
- Write operations start a timer that expires immediately
  - It is inserted in the front of the timer queue and the handler is called right away
  - Writes are thus effectively handled with higher priority than reads and waveform transfers



Starting a new timer (e. g. for a communication timeout)



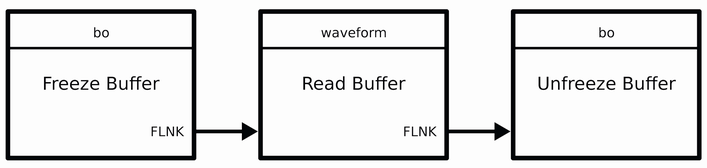
Starting a timer that expires immediately (e. g. for a write request)

Advantages

- All I/O operations are performed in the context of the timer-queue thread (no separate locking required)
- Asynchronous implementation with timers makes the code easier to test (avoids “sleep” statements that would slow down unit tests)

Waveform Read-out

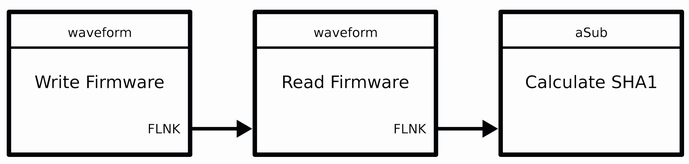
- Reading a waveform record initiates an asynchronous read
  1. The PACT flag of the waveform record is set when the record is processed
  2. The driver requests the entire array
  3. The device transmits the array chunk by chunk
  4. If a chunk is lost a timeout expires and the missing part is requested again until all blocks have been received
  5. The assembled array is passed on to Asyn
  6. Asyn clears the PACT flag and triggers processing of the record
- Devices can support a command for freezing circular buffers. This can be implemented on the IOC as a chain of records that freeze, read and unfreeze a buffer:



- A read-out mode geared towards streaming applications is under development

Firmware Update

- Users read/write firmware images through waveform records
- Processing the read/write waveform record initiates asynchronous read/write
  1. Asyn sets PACT flag before it calls write routine
  2. Data is read/written block by block (timer driven), erasing blocks before writing them
  3. When all data has been read/written, the data is passed to Asyn (Asyn clears PACT flag)
  4. When firmware write has completed, the firmware image is automatically read to update read records
- An aSub record calculates the SHA1 hash of the data read back. This hash can be compared to the output of “sha1sum firmware.bin” on the command line
  - The hash is archived as a record of firmware updates
  - In the future the hash value could be used to revoke run permit while invalid firmware is loaded



- FPGA engineers can update firmware themselves
- Access security prevents non-experts from modifying firmware

Development Tools

- Compiler: GCC/clang using C++14
- Build system: CMake, CMake4EPICS
- Code coverage report: gcov
- Unit testing framework: Google Test/Google Mock
  - Mock asynOctetSyncIO functionality to verify that the support module calls Asyn's reads/write functions correctly
  - Mock object is injected for unit tests and asynOctetSyncIO for production

Conclusion

- Support module is used successfully for LLRF controllers
- Protocol and support module are generic (device/application agnostic)
- Both scalar read/write and waveform readout are supported
- Firmware can be programmed and verified over Channel Access
- Driver handles large number of devices efficiently (largest IOC has 168 devices and 220,000 records)

FRIB Facility for Rare Isotope Beams
U.S. Department of Energy Office of Science
Michigan State University

This material is based upon work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661.
Michigan State University designs and establishes FRIB as a DOE Office of Science National User Facility in support of the mission of the Office of Nuclear Physics.